**Accessibility Document**

**Main Points and Rules**

If you are unsure or can’t find info you are looking for in this document, you can find it on the following references

<https://web.dev/>

<https://www.w3.org/TR/wai-aria-practices/>

* Try and always use native HTML elements e.g. rather use <button> vs <div>
* Make sure if using ARIA, it is used correctly. No ARIA is better than bad ARIA.
* Use headings to outline page structure
* Don’t skip heading levels
* Use landmarks to aid navigation
* Ensure correct naming and labels are given.
* When working with icon buttons (or elements with no name), use aria-label to describe the button.
* Use correct labeling of inputs by either wrapping the input in a <label>   
  <label>  
   <input type=”checkbox”>Label for Checkbox</input>  
  </label>  
  **or** by using the label’s for attribute and refer to the element’s id  
  <input id=”promo” type=”checkbox”></input>  
  <label for=”promo”>Label for checkbox</label>
* Test your development with assistive technology. ChromeVox is a free chrome plugin that you can use   
  <https://chrome.google.com/webstore/detail/chromevox-classic-extensi/kgejglhpjiefppelpmljglcjbhoiplfn?hl=en>

**Accordion / Groups**

<https://www.w3.org/TR/wai-aria-practices/#accordion>

* The title of each accordion header is contained in an element with role button.
* Each accordion header button is wrapped in an element with role heading that has a value set for aria-level that is appropriate for the information architecture of the page.
  + If the native host language has an element with an implicit heading and aria-level, such as an HTML heading tag, a native host language element may be used.
  + The button element is the only element inside the heading element. That is, if there are other visually persistent elements, they are not included inside the heading element.
* If the accordion panel associated with an accordion header is visible, the header button element has aria-expanded set to true. If the panel is not visible, aria-expanded is set to false.
* The accordion header button element has aria-controls set to the ID of the element containing the accordion panel content.
* If the accordion panel associated with an accordion header is visible, and if the accordion does not permit the panel to be collapsed, the header button element has aria-disabled set to true.
* Optionally, each element that serves as a container for panel content has role region and aria-labelledby with a value that refers to the button that controls display of the panel.
  + Avoid using the region role in circumstances that create landmark region proliferation, e.g., in an accordion that contains more than approximately 6 panels that can be expanded at the same time.
  + Role region is especially helpful to the perception of structure by screen reader users when panels contain heading elements or a nested accordion.

**Breadcrumbs**

<https://www.w3.org/TR/wai-aria-practices/#breadcrumb>

* Breadcrumb trail is contained within a navigation landmark region.
* The landmark region is labelled via aria-label or aria-labelledby.
* The link to the current page has aria-current set to page. If the element representing the current page is not a link, aria-current is optional.

**Buttons**

<https://www.w3.org/TR/wai-aria-practices/#button>

* The button has role of button.
* The button has an accessible label. By default, the accessible name is computed from any text content inside the button element. However, it can also be provided with [aria-labelledby](https://www.w3.org/TR/wai-aria-1.1/#aria-labelledby) or aria-label.
* If a description of the button's function is present, the button element has aria-describedby set to the ID of the element containing the description.
* When the action associated with a button is unavailable, the button has [aria-disabled](https://www.w3.org/TR/wai-aria-1.1/#aria-disabled) set to true.
* If the button is a toggle button, it has an aria-pressed state. When the button is toggled on, the value of this state is true, and when toggled off, the state is false.

**Checkbox**

<https://www.w3.org/TR/wai-aria-practices/#checkbox>

* The checkbox has role checkbox.
* The checkbox has an accessible label provided by one of the following:
  + Visible text content contained within the element with role checkbox.
  + A visible label referenced by the value of aria-labelledby set on the element with role checkbox.
  + aria-label set on the element with role checkbox.
* When checked, the checkbox element has state aria-checked set to true.
* When not checked, it has state aria-checked set to false.
* When partially checked, it has state aria-checked set to mixed.
* If a set of checkboxes is presented as a logical group with a visible label, the checkboxes are included in an element with role group that has the property aria-labelledby set to the ID of the element containing the label.
* If the presentation includes additional descriptive static text relevant to a checkbox or checkbox group, the checkbox or checkbox group has the property aria-describedby set to the ID of the element containing the description.

**Combobox**

<https://www.w3.org/TR/wai-aria-practices/#combobox>

* In a combobox implementing the ARIA 1.1 pattern:
  + The element that serves as the combobox container has role combobox.
  + The element with role combobox contains or owns a textbox element that has either role textbox or role searchbox.
  + When the combobox popup is visible, the combobox element contains or owns an element that has role listbox, tree, grid, or dialog.
  + If the combobox popup has a role other than listbox, the element with role combobox has aria-haspopup set to a value that corresponds to the popup type. That is, aria-haspopup is set to grid, tree, or dialog. Note that elements with role combobox have an implicit aria-haspopup value of listbox.
  + When the combobox popup is visible, the textbox element has aria-controls set to a value that refers to the combobox popup element.
* The textbox element has a value for aria-multiline of false. Note that the default value of aria-multiline is false.
* When the combobox popup is not visible, the element with role combobox has aria-expanded set to false. When the popup element is visible, aria-expanded is set to true. Note that elements with role combobox have a default value for aria-expanded of false.
* When a combobox receives focus, DOM focus is placed on the textbox element.
* When a descendant of a listbox, grid, or tree popup is focused, DOM focus remains on the textbox and the textbox has aria-activedescendant set to a value that refers to the focused element within the popup.
* In a combobox with a listbox, grid, or tree popup, when a suggested value is visually indicated as the currently selected value, the option, gridcell, row, or treeitem containing that value has aria-selected set to true.
* If the combobox has a visible label, the element with role combobox has aria-labelledby set to a value that refers to the labelling element. Otherwise, the combobox element has a label provided by aria-label.
* The textbox element has aria-autocomplete set to a value that corresponds to its autocomplete behaviour:
  + none: When the popup is displayed, the suggested values it contains are the same regardless of the characters typed in the textbox.
  + list: When the popup is triggered, it presents suggested values that complete or logically correspond to the characters typed in the textbox.
  + both: When the popup is triggered, it presents suggested values that complete or logically correspond to the characters typed in the textbox. In addition, the portion of the selected suggestion that has not been typed by the user, known as the completion string, appears inline after the input cursor in the textbox. The inline completion string is visually highlighted and has a selected state.

**Dialog**

<https://www.w3.org/TR/wai-aria-practices/#dialog>

* The element that serves as the dialog container has a role of dialog.
* All elements required to operate the dialog are descendants of the element that has role dialog.
* The dialog container element has aria-modal set to true.
* The dialog has either:
  + A value set for the aria-labelledby property that refers to a visible dialog title.
  + A label specified by aria-label.
* Optionally, the aria-describedby property is set on the element with the dialog role to indicate which element or elements in the dialog contain content that describes the primary purpose or message of the dialog. Specifying descriptive elements enables screen readers to announce the description along with the dialog title and initially focused element when the dialog opens.

**NOTE**

* Because marking a dialog modal by setting aria-modal to true can prevent users of some assistive technologies from perceiving content outside the dialog, users of those technologies will experience severe negative ramifications if a dialog is marked modal but does not behave as a modal for other users. So, mark a dialog modal **only when both:**
  1. Application code prevents all users from interacting in any way with content outside of it.
  2. Visual styling obscures the content outside of it.
* The aria-modal property introduced by ARIA 1.1 replaces aria-hidden for informing assistive technologies that content outside a dialog is inert. However, in legacy dialog implementations where aria-hidden is used to make content outside a dialog inert for assistive technology users, it is important that:
  1. aria-hidden is set to true on each element containing a portion of the inert layer.
  2. The dialog element is not a descendant of any element that has aria-hidden set to true.

**Menu**

<https://www.w3.org/TR/wai-aria-practices/#menu>

* A menu is a container of items that represent choices. The element serving as the menu has a role of either menu or menubar.
* The items contained in a menu are child elements of the containing menu or menubar and have any of the following roles:
  + menuitem
  + menuitemcheckbox
  + menuitemradio
* If activating a menuitem opens a submenu, the menuitem is known as a parent menuitem. A submenu's menu element is:
  + Contained inside the same menu element as its parent menuitem.
  + Is the sibling element immediately following its parent menuitem.
* A parent menuitem has aria-haspopup set to either menu or true.
* A parent menuitem has aria-expanded set to false when its child menu is not visible and set to true when the child menu is visible.
* One of the following approaches is used to enable scripts to move focus among items in a menu as described in § 6.6 Keyboard Navigation Inside Components:
  + The menu container has tabindex set to -1 or 0 and aria-activedescendant set to the ID of the focused item.
  + Each item in the menu has tabindex set to -1, except in a menubar, where the first item has tabindex set to 0.
* When a menuitemcheckbox or menuitemradio is checked, aria-checked is set to true.
* When a menu item is disabled, aria-disabled is set to true.
* Items in a menu may be divided into groups by placing an element with a role of separator between groups. For example, this technique should be used when a menu contains a set of menuitemradio items.
* All separators should have aria-orientation consistent with the separator's orientation.
* If a menubar has a visible label, the element with role menubar has aria-labelledby set to a value that refers to the labelling element. Otherwise, the menubar element has a label provided by aria-label.
* If a menubar is vertically oriented, it has aria-orientation set to vertical. The default value of aria-orientation for a menubar is horizontal.
* An element with role menu either has:
  + aria-labelledby set to a value that refers to the menuitem or button that controls its display.
  + A label provided by aria-label.
* If a menu is horizontally oriented, it has aria-orientation set to horizontal. The default value of aria-orientation for a menu is vertical.

**Menu Button**

<https://www.w3.org/TR/wai-aria-practices/#menubutton>

* The element that opens the menu has role button.
* The element with role button has aria-haspopup set to either menu or true.
* When the menu is displayed, the element with role button has aria-expanded set to true. When the menu is hidden, it is recommended that aria-expanded is not present. If aria-expanded is specified when the menu is hidden, it is set to false.
* The element that contains the menu items displayed by activating the button has role menu.
* Optionally, the element with role button has a value specified for aria-controls that refers to the element with role menu.
* Additional roles, states, and properties needed for the menu element are described in § 3.15 Menu or Menu bar.

**Radio Group**

<https://www.w3.org/TR/wai-aria-practices/#radiobutton>

* The radio buttons are contained in or owned by an element with role radiogroup.
* Each radio button element has role radio.
* If a radio button is checked, the radio element has aria-checked set to true. If it is not checked, it has aria-checked set to false.
* Each radio element is labelled by its content, has a visible label referenced by aria-labelledby, or has a label specified with aria-label.
* The radiogroup element has a visible label referenced by aria-labelledby or has a label specified with aria-label.
* If elements providing additional information about either the radio group or each radio button are present, those elements are referenced by the radiogroup element or radio elements with the aria-describedby property.

**Slider**

<https://www.w3.org/TR/wai-aria-practices/#slider>

* The element serving as the focusable slider control has role slider.
* The slider element has the aria-valuenow property set to a decimal value representing the current value of the slider.
* The slider element has the aria-valuemin property set to a decimal value representing the minimum allowed value of the slider.
* The slider element has the aria-valuemax property set to a decimal value representing the maximum allowed value of the slider.
* If the value of aria-valuenow is not user-friendly, e.g., the day of the week is represented by a number, the aria-valuetext property is set to a string that makes the slider value understandable, e.g., "Monday".
* If the slider has a visible label, it is referenced by aria-labelledby on the slider element. Otherwise, the slider element has a label provided by aria-label.
* If the slider is vertically oriented, it has aria-orientation set to vertical. The default value of aria-orientation for a slider is horizontal.

**Tabs**

<https://www.w3.org/TR/wai-aria-practices/#tabs>

* The element that serves as the container for the set of tabs has role tablist.
* Each element that serves as a tab has role [tab](https://www.w3.org/TR/wai-aria-1.1/#tab) and is contained within the element with role tablist.
* Each element that contains the content panel for a tab has role tabpanel.
* If the tab list has a visible label, the element with role tablist has [aria-labelledby](https://www.w3.org/TR/wai-aria-1.1/#aria-labelledby) set to a value that refers to the labelling element. Otherwise, the tablist element has a label provided by [aria-label](https://www.w3.org/TR/wai-aria-1.1/#aria-label).
* Each element with role tab has the property [aria-controls](https://www.w3.org/TR/wai-aria-1.1/#aria-controls) referring to its associated tabpanel element.
* The active tab element has the state aria-selected set to true and all other tab elements have it set to false.
* Each element with role tabpanel has the property [aria-labelledby](https://www.w3.org/TR/wai-aria-1.1/#aria-labelledby) referring to its associated tab element.
* If a tab element has a pop-up menu, it has the property aria-haspopup set to either menu or true.
* If the tablist element is vertically oriented, it has the property aria-orientation set to vertical. The default value of aria-orientation for a tablist element is horizontal.

**Tree View**

<https://www.w3.org/TR/wai-aria-practices/#TreeView>

* All tree nodes are contained in or owned by an element with role tree.
* Each element serving as a tree node has role treeitem.
* Each root node is contained in the element with role tree or referenced by an aria-owns property set on the tree element.
* Each parent node contains or owns an element with role group.
* Each child node is contained in or owned by an element with role group that is contained in or owned by the node that serves as the parent of that child.
* Each element with role treeitem that serves as a parent node has aria-expanded set to false when the node is in a closed state and set to true when the node is in an open state. End nodes do not have the aria-expanded attribute because, if they were to have it, they would be incorrectly described to assistive technologies as parent nodes.
* If the tree supports selection of more than one node, the element with role tree has aria-multiselectable set to true. Otherwise, aria-multiselectable is either set to false or the default value of false is implied.
* If the tree does not support multiple selection, aria-selected is set to true for the selected node and it is not present on any other node in the tree.
* if the tree supports multiple selection:
  + All selected nodes have aria-selected set to true.
  + All nodes that are selectable but not selected have aria-selected set to false.
  + If the tree contains nodes that are not selectable, those nodes do not have the aria-selected state.
* The element with role tree has either a visible label referenced by aria-labelledby or a value specified for aria-label.
* If the complete set of available nodes is not present in the DOM due to dynamic loading as the user moves focus in or scrolls the tree, each node has aria-level, aria-setsize, and aria-posinset specified.
* If the tree element is horizontally oriented, it has aria-orientation set to horizontal. The default value of aria-orientation for a tree is vertical.

**NOTE**

If aria-owns is set on the tree container to include elements that are not DOM children of the container, those elements will appear in the reading order in the sequence they are referenced and after any items that are DOM children. Scripts that manage focus need to ensure the visual focus order matches this assistive technology reading order.

**Tree Grid**

<https://www.w3.org/TR/wai-aria-practices/#treegrid>

* The treegrid container has role treegrid.
* Each row container has role row and is either a DOM descendant of or owned by the treegrid element or an element with role rowgroup.
* Each cell is either a DOM descendant of or owned by a row element and has one of the following roles:
  + columnheader if the cell contains a title or header information for the column.
  + rowheader if the cell contains title or header information for the row.
  + gridcell if the cell does not contain column or row header information.
* A row that can be expanded or collapsed to show or hide a set of child rows is a parent row. Each parent row has the aria-expanded state set on either the row element or on a cell contained in the row. The aria-expanded state is set to false when the child rows are not displayed and set to true when the child rows are displayed. Rows that do not control display of child rows do not have the aria-expanded attribute because, if they were to have it, they would be incorrectly described to assistive technologies as parent rows.
* If the treegrid supports selection of more than one row or cell, it is a multi-select treegrid and the element with role treegrid has aria-multiselectable set to true. Otherwise, it is a single-select treegrid, and aria-multiselectable is either set to false or the default value of false is implied.
* If the treegrid is a single-select treegrid, aria-selected is set to true on the selected row or cell, and it is not present on any other row or cell in the treegrid.
* if the treegrid is a multi-select treegrid:
  + All selected rows or cells have aria-selected set to true.
  + All rows and cells that are not selected have aria-selected set to false.
* If there is an element in the user interface that serves as a label for the treegrid, aria-labelledby is set on the grid element with a value that refers to the labelling element. Otherwise, a label is specified for the grid element using aria-label.
* If the treegrid has a caption or description, aria-describedby is set on the grid element with a value referring to the element containing the description.
* If the treegrid provides sort functions, aria-sort is set to an appropriate value on the header cell element for the sorted column or row as described in the section on grid and table properties.
* If the treegrid provides content editing functionality and contains cells that may have edit capabilities disabled in certain conditions, aria-readonly is set to true on cells where editing is disabled. If edit functions are disabled for all cells, instead of setting aria-readonly to true on every cell, aria-readonly may be set to true on the treegrid element. Treegrids that do not provide cell content editing functions do not include the aria-readonly attribute on any of their elements.
* If there are conditions where some rows or columns are hidden or not present in the DOM, e.g., data is dynamically loaded when scrolling or the grid provides functions for hiding rows or columns, the following properties are applied as described in the section on grid and table properties.
  + aria-colcount or aria-rowcount is set to the total number of columns or rows, respectively.
  + aria-colindex or aria-rowindex is set to the position of a cell within a row or column, respectively.
* If the treegrid includes cells that span multiple rows or multiple columns, and if the treegrid role is NOT applied to an HTML table element, then aria-rowspan or aria-colspan is applied as described in grid and table properties.

**NOTE**

* A treegrid built from an HTML table that includes cells that span multiple rows or columns must use HTML rowspan and colspan and must not use aria-rowspan or aria-colspan.
* If rows or cells are included in a treegrid via aria-owns, they will be presented to assistive technologies after the DOM descendants of the treegrid element unless the DOM descendants are also included in the aria-owns attribute.

**Landmark Regions**

**Steps for landmark regions**

* Identify logical structure
* Assign landmark roles to each area
* Label areas

Landmark Roles

**Banner**

If the HTML header element technique is not being used, a role="banner" attribute should be used to define a banner landmark.

**Complementary**

If the HTML aside element technique is not being used, use a role="complementary" attribute to define a complementary landmark.

**Contentinfo**

If the HTML footer element technique is not being used, a role="contentinfo" attribute should be used to define a contentinfo landmark.

**Form**

Use the role="form" to identify a region of the page; do not use it to identify every form field.

**Main**

If the HTML main element technique is not being used, use a role="main" attribute to define a main landmark.

**Navigation**

If the HTML nav element technique is not being used, use a role="navigation" attribute to define a navigation landmark.

**Region**

If the HTML section element technique is not being used, use a role="region" attribute to define a region landmark.

**Search**

The role="search" attribute defines a search landmark.